Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____

Кафедра _____Програмної інженерії_____

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів обробки великих даних в контекстнозалежній системі повнотекстового пошуку публікацій

Виконав: студент 2 курсу, групи ІПЗм-18-1___

_____Поляков Я.С._____
(прізвище, ініціали)

спеціальності 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

Освітньо-наукової програми_____
(тип програми)

Інженерія програмного забезпечення
(повна назва освітньої програми)

Керівник _____д.т.н., проф. Четвериков Г.Г. _____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри, проф. _____ З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет_Комп'ютерних наук_____

Кафедра_Програмної інженерії_____

Рівень вищої освіти - другий (магістерський)

Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)

Тип програми освітньо-наукова програма_____

Освітня програма_Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Полякову Ярославу Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів обробки великих даних в контекстнозалежній системі повнотекстового пошуку публікацій_____
затверджена наказом університету від "___" _____2020 р № _____
2. Термін подання студентом роботи до екзаменаційної комісії
_____
3. Вихідні дані до роботи методи обчислення великих даних, архітектура системи для обчислення великих даних з використанням хмарних технологій, пояснювальна записка.
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз проблемної галузі і постановка задачі, аналіз методів обробки великих даних, методи доступу та управління сховищами даних, архітектура розгортання системи в хмарних середовищах

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. Четвериков Г.Г.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	15 лютого 2020р.	
2.	Огляд варіантів систем, що використовують методи великих даних	15 березня 2020р.	
3.	Моделювання архітектури великих даних	17 березня 2020р.	
4.	Підготовка пояснювальної записки	28 березня 2020р.	
5.	Спецчастина	19 квітня 2020р.	
6.	Підготовка презентації та доповіді	06 травня 2020р.	
7.	Попередній захист	травня 2020р.	
8.	Нормоконтроль, рецензування	травня 2020р.	
9.	Занесення диплома в електронний архів	травня 2020р.	
10.	Допуск до захисту у зав. кафедри	травня 2020р.	

Дата видачі завдання _____ _____2020 р.

Студент _____
(підпис)

Керівник роботи _____ д.т.н., проф. Четвериков Г.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи магістра, 39 с., 12 рисунків, 15 джерел.

АТЕСТАЦІЙНА РОБОТА, ОБЧИСЛЕННЯ ВЕЛИКИХ ДАННИХ, ХМАРНЕ СЕРЕДОВИЩЕ, МІКРОСЕРВІСИ, КЛАСТЕР.

Об’єктом дослідження є обробка великих даних. Предметом дослідження є методи моделювання архітектури для обробки великих даних.

Метою роботи є дослідження методів обробки великих даних та розгортання використовуючи хмарну архітектуру.

У результаті роботи було проведені дослідження методів використання обробки великих даних розраховані на роботу з великими навантаженнями з використанням хмарних середовищ. Було проведене моделювання та порівняння методів роботи з великими об’ємами даними в хмарному середовищі з високими навантаженнями.

ATTESTATION WORK, BIG DATA CALCULATION, CLOUD ENVIRONMENT, MICROSERVISES, CLUSTER.

The object of the study is to process big data. The subject of the study is methods of modeling architecture for processing big data.

The purpose of the study is to investigate Big Data processing and deployment methods using a cloud architecture.

As a result of the research were conducted methods of using big data processing designed to work with high loads using cloud environments. The modeling and comparison of large data volumes in high-load cloud environments was performed.

ЗМІСТ

ВСТУП

В сучасному світі для полегшення та автоматизації людської діяльності з’являється потреба працювати з великими масивами даних. Існує проблема зі швидкістю збирання, обчислення великих об’ємів інформації. Саме через це все стало складніше створювати системи та сервери, архітектура яких допомогла б працювати з такою кількістю даних. Через попит на обчислення величезних об’ємів даних існує великий вибір хмарних технологій, які дають можливість створювати складні системи в хмарному оточенні.

Подібні системи для роботи з великим обсягом даних є дуже перспективними, адже аналізуючи дані, можна отримати дослідити перспективи того, чи іншого продукту, прогнозувати ринок та поведінку клієнтів. Подібні системи використовуються в маркетингу, фінансах, телекомунікаціях, рітейлі, енергетичній промисловості, державному секторі (все, що стосується електронного урядування) і так далі.

Маючи велику кількість методів та технологій для організації роботи з великими даних є можливості створювати системи для збору та аналізу даних у різних галузях, що має дати змогу покращити ведення справ у будь-яких доцільних сферах життя.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Принципи та методи обробки великих даних

Термін “Великі дані” означає щось більше, ніж просто аналіз величезних обсягів інформації. Проблема не в тому, що поставлена задача обробити величезні обсяги даних, а в тому, що більша їх частина представлена не в структурованому форматі, які не відповідають традиційнім структурованим форматам. Все це зберігається в безлічі різноманітних сховищ. В результаті можна мати доступ до величезного обсягу даних і не мати необхідних інструментів, щоб встановити взаємозв'язок між цими даними і зробити на їх основі якісь висновки. Усвідомлюючи той факт, що дані зараз оновлюються все частіше і частіше, складеться ситуація, в якій традиційні методи аналізу інформації не можуть наздогнати величезними обсягами постійно оновлюваних даних, що в підсумку і відкриває дорогу технологіям великих даних.

Визначальними характеристиками для великих даних є, окрім їх фізичного об’єму, й інші, які підкреслюють складність задачі обробки і аналізу цих даних. Набір даних VVV, (volume, velocity, variety - фізичний об’єм, швидкість приросту даних і необхідність їх швидкої обробки, здатність обробляти дані різних типів) був розроблений з метою вказати на рівну значимість управління даними по всім трьом аспектам.

У подальшому з’явилась інтерпретація з чотирьома V (додалась veracity - достовірність), п’яти V (viability - життєздатність і value - цінність), семи V (variability - змінність та visualization - візуалізація).

Виходячи з вищеназваних визначень, основні принципи роботи з великими даними такі:

- горизонтальна масштабованість. Це є базовим принципом обробки великих даних. Як вже було зазначено, великих даних з кожним днем стає все

більше. Відповідно, необхідно збільшувати кількість обчислювальних вузлів, за якими розподіляються ці дані, при чому обробка має відбуватись без погіршення продуктивності;

- відмовостійкість. Цей принцип витікає з попереднього. Оскільки обчислювальних вузлів у кластері може бути багато (іноді десятки тисяч) та їх кількість, не виключено, буде збільшуватись, зростає ймовірність виходу машин з ладу. Методи роботи з великими даними мають враховувати ймовірність таких ситуацій і передбачати превентивні заходи;
- локальність даних. Оскільки дані розподілені по великій кількості обчислювальних вузлів, то, якщо вони фізично знаходяться на одному сервері, а обробляються на іншому, витрати на передачу даних можуть бути невиправдано великими. Тому обробку даних бажано проводити на тій же машині, на якій вони зберігаються.

Ці принципи відрізняються від тих, які характерні для традиційних, централізованих, вертикальних моделей зберігання структурованих даних.

В стек підходів і технологій включають засоби масово-паралельної обробки невизначено-структурованих даних, такі як NoSQL бази даних, алгоритми MapReduce і засоби проекту Hadoop. У подальшому до технологій великих даних почали відносити й інші рішення, що забезпечують схожі за характеристиками можливості обробки надвеликих масивів даних, а також деякі апаратні засоби.

MapReduce - модель розподілених обчислювань у комп’ютерних кластерах. Згідно з цією моделлю, додаток розділяється на значну кількість однакових елементарних завдань, що виконуються на вузлах кластера і потім, природнім шляхом зводяться у кінцевий результат.

NoSQL - загальний термін для різних нереляційних баз даних і сховищ, не означає якусь конкретну технологію чи продукт. Звичайні реляційні бази даних добре підходять для досить швидких і однотипних запитів, а на складних і гнучко побудованих запитах, характерних для великих даних, навантаження перевищує розумні межі і використання СУБД стає неефективним.

Hadoop - набір інструментів, бібліотек і фреймворків, що вільно розповсюджується, для розробки і виконання розподілених програм, які працюють на кластерах із сотень і тисяч вузлів. Вважається однією з основоположних технологій більшості даних.

R - мова програмування для статистичної обробки даних і роботи з графікою. Широко використовується для аналізу даних і фактично став стандартом для статистичних програм.

Виділяються наступні методи і техніки аналізу, що застосовуються до великих даних:

- методи класу Data Mining (видобуток даних, інтелектуальний аналіз даних, глибинний аналіз даних) - сукупність методів виявлення у даних раніше невідомих, нетривіальних, практично корисних знань, необхідних для прийняття рішень. До таких методів, зокрема, належать: навчання асоціативним правилам (association rule learning), класифікація (разгалуження на категорії), кластерний аналіз, регресійний аналіз, виявлення і аналіз відхилень;
- краудсорсинг - класифікація і збагачення даних силами широкого, неозначеного кола особистостей, що виконують цю роботу без вступу у трудові стосунки;
- змішання та інтеграція даних - набір технік, що дозволяють інтегрувати різнорідні дані з розмаїття джерел з метою проведення глибинного аналізу (наприклад, цифрова обробка сигналів, обробка природньої мови, включно з тональним аналізом) ;
- машинне навчання, включаючи навчання з учителем і без учителя – використання моделей, побудованих на базі статистичного аналізучи машинного навчання для отримання комплексних прогнозів на основі базових моделей;
- штучні нейронні мережі, мережевий аналіз, оптимізація, у тому числі генетичні алгоритми;

- розпізнавання образів;
- прогнозна аналітика;
- імітаційне моделювання (simulation) - метод, що дозволяє будувати моделі, що описують процеси так, як вони би проходили у дійсності. Імітаційне моделювання модна розглядати як різновид експериментальних випробувань;
- просторовий аналіз (spatial analysis) - клас методів, що використовують топологічну, геометричну і географічну інформацію, що вилучається із даних;
- статистичний аналіз - аналіз часових рядів, A/B-тестування, при його використанні контрольна група елементів порівнюється із набором тестових груп, у яких один чи кілька показників були змінені, щоб з’ясувати, які зі змін покращують цільовий показник;
- візуалізація аналітичних даних - подання інформації у вигляді малюнків, діаграм, з використанням інтерактивних можливостей і анімації, як для отримання результатів, так і для використання у якості вихідних даних для подальшого аналізу. Дуже важливий етап аналізу великих даних, що дозволяє показати найважливіші результати аналізу у найбільш зручному для сприйняття вигляді.

Метод MapReduce займає фундаментальне значення у роботі з великими даними. Вхідні дані зазвичай мають великий об'єм, і обчислення доводиться розподіляти по сотнях або навіть тисячах машин, щоб мати можливість вирішити завдання в розумний проміжок часу. Проблеми розпаралелювання обчислень, розподіли даних і обробки збоїв примушують відмовитися від простої моделі обчислень з великим об'ємом складного коду. Ця абстракція дозволяє легко робити необхідні обчислення, але приховує усі деталі розпаралелювання, відмовостійкості, розподілу даних і балансування навантаження у своїх бібліотеках.

MapReduce складається із стадій Map, Shuffle і Reduce. Як правило, в практичних завданнях найважчою виявляється стадія Shuffle, оскільки на цій стадії відбувається сортування даних. Насправді існує ряд завдань, в яких можна обійтися тільки стадією Map.

Метод Map передає вхідні дані вирішуваної задачі представляють великий список значень, і на кроці Map відбувається його попередня обробка. Для цього головний вузол кластера (master node) отримує цей список, ділить його на частини і передає робочим вузлам (worker node), принцип роботи метода зображено на рисунку 1.1.

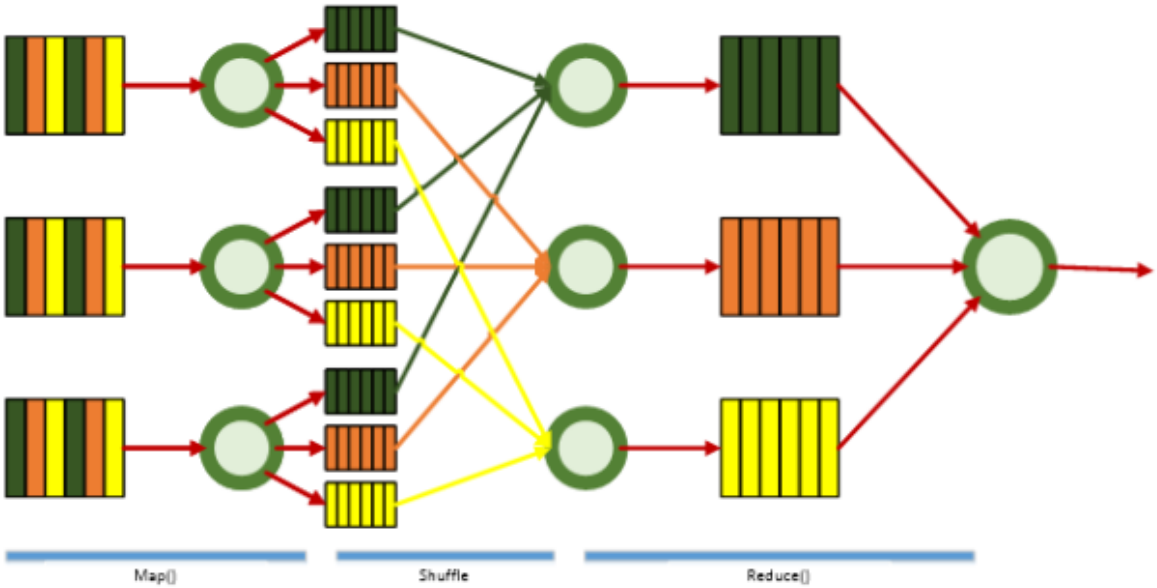

Рисунок 1.1 – Модель роботи метода MapReduce

Кожен робочий вузол застосовує функцію Map до локальних даних і записує результат у форматі "ключ-значення" в тимчасове сховище.

Shuffle: робочі вузли перерозподіляють дані на основі ключів, створених функцією Map, так, що усі дані належать одному ключу котрі лежать на одному робочому вузлі.

Reduce: робочі вузли обробляють кожну групу результатів по порядку дотримання ключів. Головний вузол отримує проміжні відповіді від робочих

вузлів і передає їх на вільні вузли для виконання наступного кроку. Що вийшов після проходження усіх необхідних кроків результат - це рішення задачі, яка спочатку формулювалася.

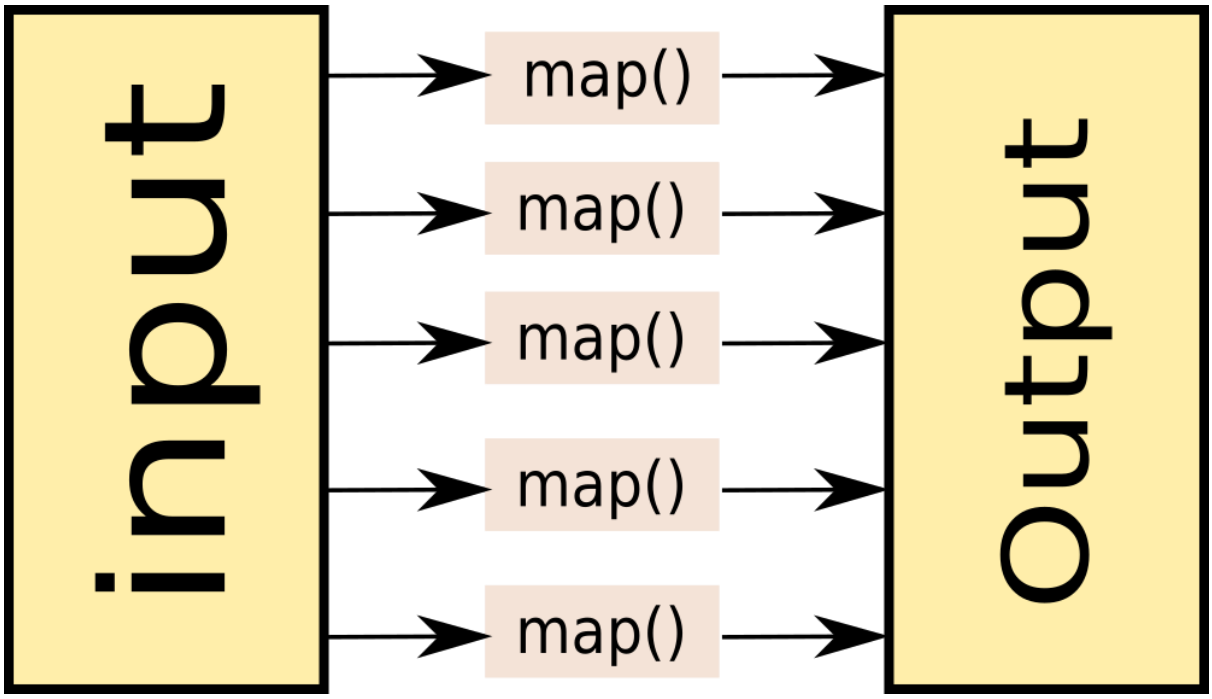

Рисунок 1.2 – Модель роботи Map Reduce

Етап фільтрації даних вирішується за допомогою методів Map (рис.1.2). При створенні Map - Only завдання в Hadoop треба вказати нульову кількість Reducer.

1.2 Особливості організації архітектури великих даних

Архітектура для обробки великих даних дозволяє приймати, обробляти і аналізувати дані, які є занадто об'ємними або занадто складними для традиційних систем баз даних. Час, коли організації починають використати великі дані, залежить від можливостей користувачів і їх засобів. Для деяких це можуть бути сотні гігабайт даних, а для інших сотні терабайт. У міру вдосконалення засобів

для роботи з великими наборами даних, змінюється і значення великих даних. Часто цей термін пов'язаний зі значенням, яке можна витягнути з наборів даних за допомогою розширеної аналітики, а не виключно з розміром даних. Хоча в цих випадках вони зазвичай досить великі.

З плином часу ландшафт даних змінився. Крім того, з'являється все більше нових можливостей для роботи з даними. Об’єми сховищ тільки збільшуються в геометричній прогресії, все більше з’являється постачальників хмарних технологій, щоб задовольнити попит на обробку та зберігання інформації. Дані поступають в прискореному темпі, їх постійно треба збирати і переглядати. Інші дані поступають повільніше, але в дуже великих блоках. Вони часто містять дані журналів за десятиліття. Існує ймовірність зійтися з проблемою розширеної аналітики або проблемою, для вирішення якої вимагається використати машинне навчання, це завдання, які архітектура для обробки великих даних призначена вирішити.

Рішення для обробки великих даних зазвичай призначені для одного або декількох з наступних типів робочого навантаження :

- пакетна обробка джерел неактивних великих даних;
- обробка великих даних в динаміці в режимі реального часу;
- інтерактивне вивчення великих даних;
- прогнозна аналітика і машинне навчання.

Використайте архітектуру для обробки великих даних для наступних сценаріїв:

- зберігання і обробка даних в об'ємах, занадто великих для традиційної бази даних;
- перетворення неструктурованих даних для аналізу і створення звітів;
- запис, обробка і аналіз неприв'язаних потоків даних в режимі реального часу або з низькою затримкою.

На схемі нижче (див. рис. 1.1) показані логічні компоненти, які входять в архітектуру для обробки великих даних.

Більшість архітектури для обробки великих даних включають деякі або усі перераховані нижче компоненти (див. рис. 1.3).

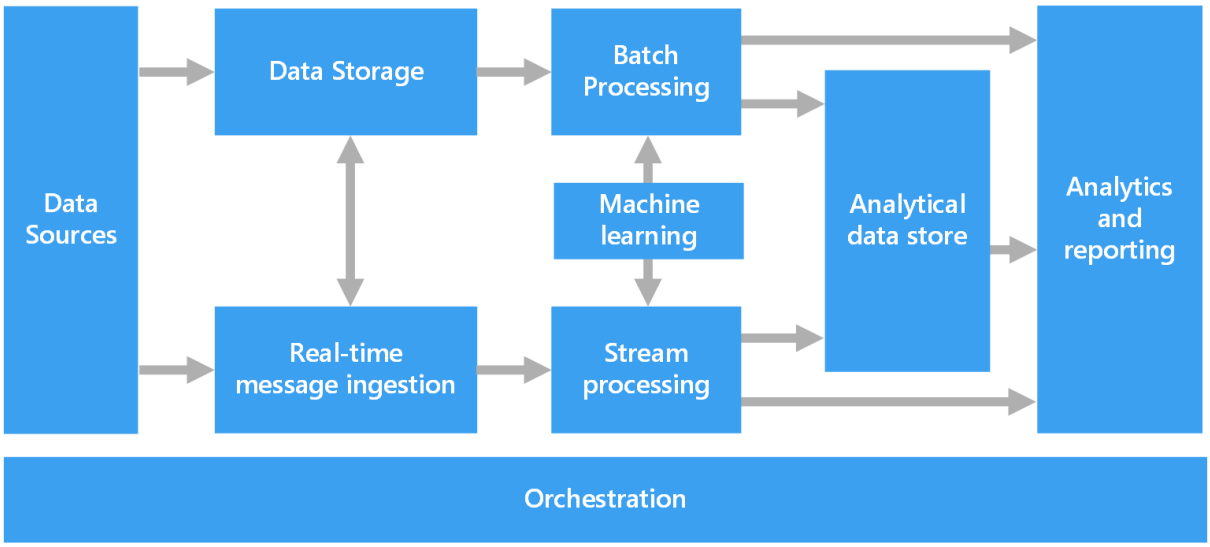

Рисунок 1.3– Компоненти архітектури великих даних

Сховище даних. Дані для пакетної обробки зазвичай зберігаються в розподіленому сховищі файлів, де можуть міститися значні об'єми великих файлів в різних форматах. Цей тип сховища часто називають озером даних. Таке сховище можна реалізувати за допомогою Azure Data Lake Store або контейнерів великих двійкових об'єктів в службі сховища Azure.

Прийом повідомлень в реальному часі. Якщо рішення містить джерела в режимі реального часу, в архітектурі має бути передбачений спосіб збору і збереження повідомлень в режимі реального часу для потокової обробки. Це може бути просте сховище даних з текою, в яку повідомлення, що входять, поміщаються для обробки. Але для прийому повідомлень багатьох рішень потрібно сховище, яке можна використати в якості буфера. Таке сховище повинне підтримувати обробку з горизонтальним масштабуванням, надійну доставку і

іншу семантику черги повідомлень. Ця частина архітектури потокової передачі часто називається потоковою буферизацією.

Пакетна обробка. Оскільки набори даних дуже великі, часто в рішенні обробляються тривалі пакетні завдання. Для них виконується фільтрація, статистична обробка і інші процеси підготовки даних до аналізу. Зазвичай в ці завдання входить читання початкових файлів, їх обробка і запис вихідних даних в нові файли. Варіанти: виконання завдань AWS Batch, використання призначених для користувача завдань Hive, Pig або MapReduce в кластері Hadoop і застосування програм Java, Scala або Python в кластері Spark.

Потокова обробка. Зберігши повідомлення, що поступають в режимі реального часу, система виконує для них фільтрацію, статистичну обробку і інші процеси підготовки даних до аналізу. Потім оброблені потокові дані записуються у вихідний приймач. AWS Stream Analytics надає керовану службу потокової обробки на основі постійного виконання запитів SQL для неприв'язаних потоків. Крім того, для потокової передачі можна використати технології Apache з відкритим кодом, наприклад Storm і Spark Streaming в кластері HDInsight.

Сховище аналітичних даних. У багатьох рішеннях для обробки великих даних дані готуються до аналізу. Потім оброблені дані структуруються відповідно до формату запитів для засобів аналітики. Сховище аналітичних даних, використовуване для обробки таких запитів, може бути реляційною базою даних типу Kimball, як можна побачити у більшості традиційних рішень бізнес-аналітики. Крім того, дані можна представити за допомогою технології NoSQL з низькою затримкою, такий як HBase або інтерактивна база даних Hive, яка надає абстракцію метаданих для файлів даних в розподіленому сховищі. Azure Synapse Analytics - це керована служба для зберігання великих об'ємів даних в хмарі. HDInsight підтримує Interactive Hive, HBase і Spark SQL, які також можна використати, щоб надавати дані для аналізу.

Оркестрація. Більшість рішень для обробки великих даних складаються з робітників процесів, під час яких перетворяться початкові дані, що

повторюються, дані переміщаються між декількома джерелами і приймачами, оброблені дані завантажуються в сховища аналітичних даних або ж результати передаються безпосередньо в звіт або на панель моніторингу. Щоб автоматизувати ці робочі процеси, ви можете використати технологію оркестрації, таку як фабрика даних Azure або Apache Oozie і Sqoop.

1.3 Особливості Лямбда- архітектури

При роботі з дуже великими наборами даних виконання типу запитів, необхідних клієнтам, може зайняти багато часу. Ці запити не можна виконати в режимі реального часу. Вони часто вимагають алгоритмів, наприклад MapReduce, які працюють паралельно по усьому набору даних. Результати потім зберігаються окремо від необроблених даних і використовуються для виконання запитів.

Недоліком цього підходу є те, що з'являється затримка. Якщо обробка займає декілька годин, запит може повертати результати, які були актуальними кілька годин тому. У ідеалі вам слід отримати деякі результати в режимі реального часу (можливо, з деякою втратою точності) і об'єднати їх з результатами пакетної аналітики.

Лямбда-архітектура, усуває цю проблему шляхом створення двох шляхів для потоку даних (див. рис. 1.4). Усі дані, що поступають в систему, проходять через ці два шляхи:

- на пакетному рівні усі дані, що входять, зберігаються в необробленому вигляді і виконується їх пакетна обробка. Результати цієї обробки зберігаються в пакетному уявленні;
- на рівні прискорення дані аналізуються в режимі реального часу. Цей рівень забезпечує мінімальну затримку, хоча і за рахунок точності.

Пакетний рівень надає результати для рівня обслуговування, який індексує пакетне представлення для ефективного виконання запитів. Рівень прискорення оновлює рівень обслуговування, відправляючи додаткові оновлення з урахуванням останніх даних.

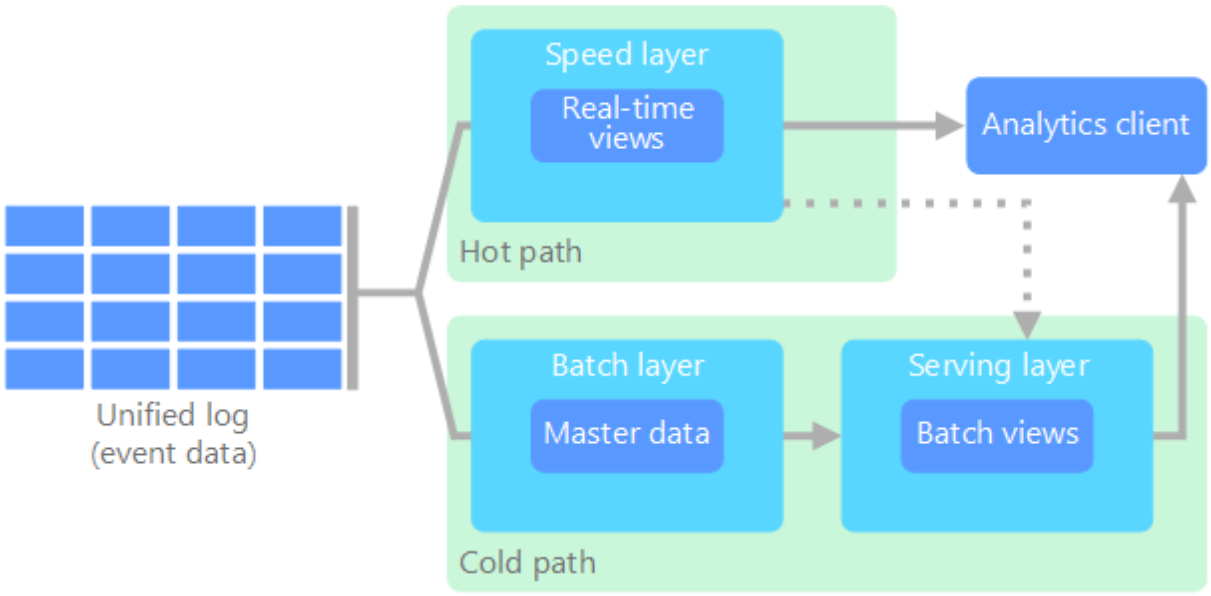

Рисунок 1.4 – Модель Лямбда-архітектури

Дані, які поступають в критичну дорогу, обмежені вимогами до затримки, накладеними рівнем прискорення, щоб їх можна було обробити якнайшвидше. Часто в цьому випадку слід забезпечити компроміс: деяка втрата точності на користь отримання готових даних якнайшвидше. Наприклад, розглядаючи сценарій Інтернету речей, де велика кількість датчиків температури відправляє ці телеметрії.

В результаті критичний і холодний шляху об'єднуються в клієнтському додатку аналітики. Якщо клієнт повинен відображати результати своєчасно, але потенційно з менш точними даними в режимі реального часу, він отримуватиме результати з критичного шляху. Інакше він вибиратиме результати з холодного шляху, щоб відображати точніші дані, проте не так своєчасно. Іншими словами, критичний шлях містить дані за відносно невеликий проміжок часу, після якого результати можна оновити точнішими даними з критичного шляху.

Необроблені дані, які зберігаються на пакетному рівні, є незмінними. Дані, що входять, завжди додаються до наявних. Попередні дані ніколи не перезаписуються. Будь-які зміни значення певних даних зберігаються у вигляді нового запису про подію з міткою часу. Це дозволяє у будь-який момент часу виконати повторне обчислення в журналі зібраних даних.

Можливість повторного обчислення пакетних представлень з початкових необроблених даних дуже важлива, оскільки це дозволяє створювати представлення у міру розвитку системи.

1.4 Особливості Каппа-архітектура

Недоліком лямбда-архітектури є її складність. Логіка обробки застосовується в двох різних місцях, в холодному і критичному шляхах, з використанням різних структур. Це призводить до дублювання логіки обчислень і ускладнює управління архітектурою для обох шляхів.

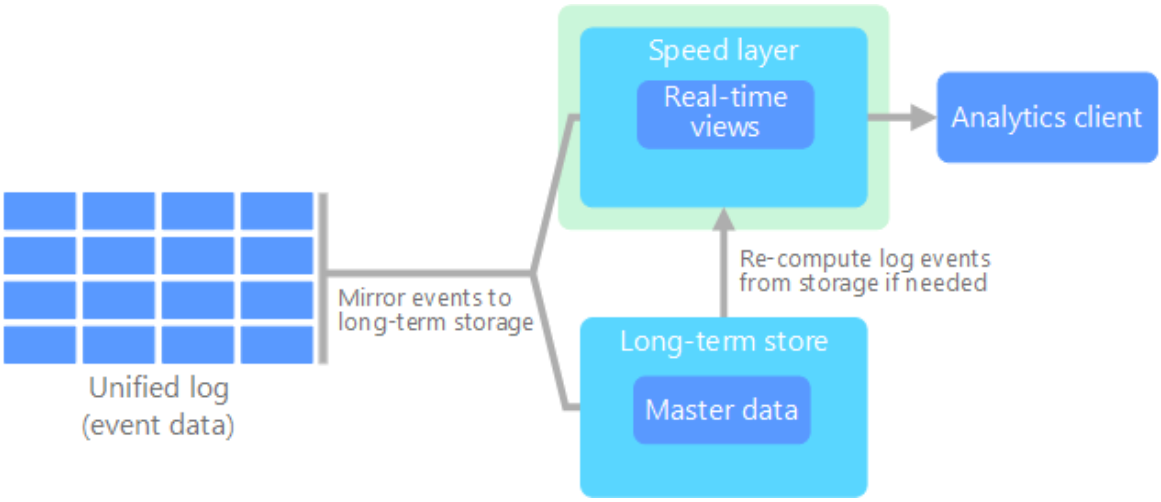

Рисунок 1.5 – Модель Капа-архітектури

Каппа-архітектура має такі ж основні цілі, що і лямбда-архітектура, але з однією важливою відмінністю: використовується система обробки потокових даних і усі дані проходять по одному шляху (див. рис. 1.5).

Є деяка подібність з пакетного рівня лямбда-архітектури. Воно полягає в тому, що дані є незмінними. Крім того, збираються усі дані, а не тільки їх підмножина. Дані приймаються як потік подій в розподіленому і відмовостійкому єдиному журналі. Ці події упорядковуються, і поточний стан події змінюється тільки при додаванні нової події. Аналогічно рівню прискорення лямбда-архітектури уся обробка подій виконується у вхідному потоці і зберігається як представлення в режимі реального часу.

Якщо необхідно повторно вичислити увесь набір даних (аналогічно тому, що відбувається на пакетному рівні в лямбда-архітектурі), просто відтворіть потік. Щоб завершити обчислення вчасно, зазвичай використовується паралелізм.

1.5 Особливості архітектури інтернет речей

З практичної точки зору Інтернет речей представляє усі пристрої, підключені до Інтернету. Сюди входять ПК, мобільний телефон, розумний годинник, розумний термостат, розумний холодильник, підключений автомобіль, імпланти для кардіомоніторинга і усі інші пристрої, які підключені до Інтернету і відправляють або отримують дані. Кількість підключених пристроїв росте щодня, як і об'єм збираних з них даних. Часто ці дані збираються в середовища з великою кількістю обмежень, а іноді і з високою затримкою. У інших випадках дані вирушають з середовищ з малою затримкою тисячами або мільйонами пристроїв, вимагаючи можливості швидко приймати дані і обробляти їх відповідним чином. Так, щоб працювати з цими обмеженнями і унікальними вимогами, потрібно продумане планування.

Керована подіями архітектура дуже зручна при роботі з рішеннями Інтернет речей. На рисунку 1.6 представлені можливі варіанти логічної архітектури для Інтернету речей. Особлива увага в цій схемі приділяється компонентам архітектури для потокової передачі подій.

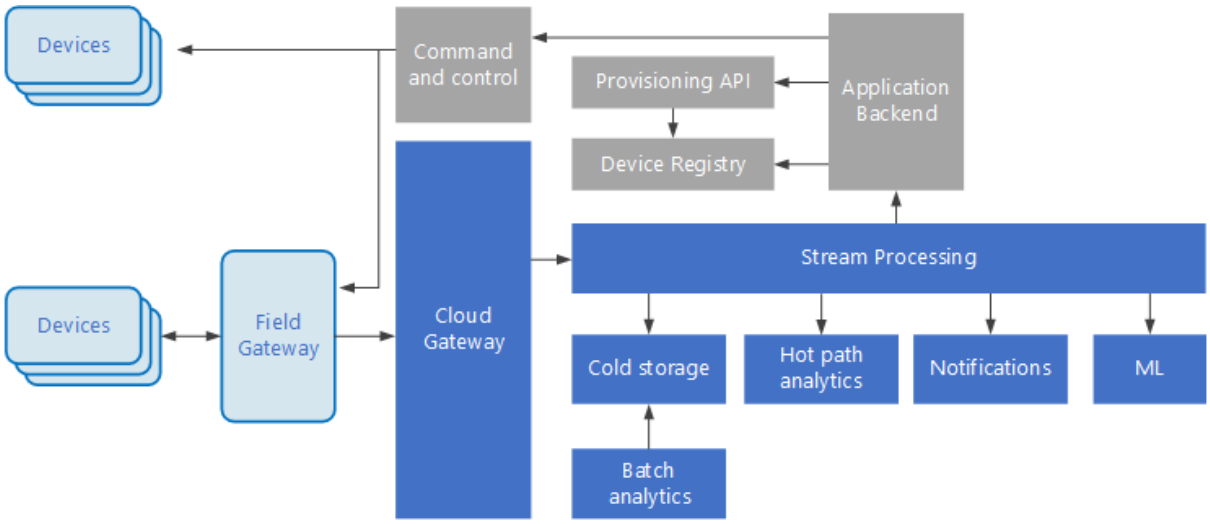

Рисунок 1.6 – Модель архітектури Інтернет речей

Хмарний шлюз приймає події від пристроїв на межі хмари, використовуючи надійну службу сполучень з низькою затримкою.

Пристрої можуть відправляти події в хмарний шлюз безпосередньо або через польовий шлюз. Польовий шлюз - цей спеціальний пристрій або програма, що зазвичай розміщуються поряд з пристроями, які отримують події і пересилають їх в хмарний шлюз.

Польовий шлюз може виконувати деяку попередню обробку подій, що збираються з пристроїв, наприклад фільтрацію, статистичну обробку або перетворення протоколів.

Отримані події проходять через один або декілька обробників потоку, які передають дані в інші системи (наприклад, сховище даних) або виконують аналітичну або іншу обробку.

Нижче наводяться приклади типових процесів обробки:

- збереження даних про події в сховищі для архівації або пакетної аналітик;
- аналітика критичного шляху, тобто аналіз потоку подій майже в режимі реального часу для виявлення аномалій, виявлення закономірностей в плаваючому діапазоні часу або створення сповіщень при виконанні певних умов в потоці;
- обробка спеціальних типів повідомлень, що не відносяться до телеметрії, наприклад повідомлень і тривожних сигналів;
- машинне навчання.

Сірі блоки означають компоненти системи Інтернету речей, не пов'язані безпосередньо з потоковою передачею подій. Вони включені в схему для повноти представлення.

Реєстр пристроїв - це база даних про підготовлені пристрої, яка містить ідентифікатори пристроїв і деякі метадані, наприклад розташування.

API підготовки - це загальний зовнішній інтерфейс для підготовки і реєстрації нових пристроїв.

У деяких рішеннях Інтернету речей допускається відправка повідомлень, що управляють, на пристрої.

1.6 Постановка задачі

Базуючись на дослідженні існуючих методів, що використовуються в великих даних, на перевагах та недоліках архітектури, можна описати певні вимоги до побудови системи розподілених обчислень з великою кількістю даних.

При обчисленні великих масивів даних використовуючи метод MapReduce ми отримуємо дуже швидкі показники в обчисленні, це досягається методом розпаралеленням на окремі вузли, тобто при створенні потоку для обчислення всі розрахунки будуть відбуватися на окремих серверах. Всі вузли по результатам

обчислень передають вихідні дані на Master Node, це забезпечує високу гнучкість і масштабованість, міняючи кількість вузлів в кластері ми отримаємо відповідні швидкісні показники в обробці даних.

Для розгортання архітектури великих даних необхідно мати велику кількість ресурсів для обчислення, які б дозволили вертикальну та горизонтальну масштабованість. Маючи такий тип архітектури, необхідно забезпечити надійне середовище з автоматичною масштабованістю та розподіленням ресурсів в залежності від кількості вхідних даних.

В архітектурі великих даних необхідно використовувати доцільні інструменти для виконування тих чи інших задач, які би були розроблені з урахуванням їх використання в роботі з великими даними. Вибираючи певний інструмент необхідно звернути увагу на можливість працювати з даними паралельно та гнучко керувати ресурсами, створювати додаткові вузли утворюючи кластери для ефективного обчислення інформації та централізовано їх збирати.

У великомасштабних мережах, де працюють одночасно багато пристроїв та серверів, стратегії розподілення даних здатні позбутися обмежень сховища даних шляхом розповсюдження даних рівномірно між усіма вузлами. Оскільки сервери в цих вузлах можуть раптово вийти з ладу, то необхідно додатково впроваджувати методи реплікації даних між серверами та сховищем даних та дотримуватись правил консистентності даних у системі. Для того, щоб підтримувати високу швидкість обробки та обчислення даних у всій інфраструктурі та в кожній локальній місцевості окремого вузла, потрібно впровадити алгоритми врівноваження навантаження.

Задачею даної роботи є моделювання системи, побудованої з використанням архітектури великих даних та дослідження методів застосування такої архітектури при роботі зі сховищами даних, а також опис переваг та недоліків її використання у порівнянні з звичайним обчисленням. Необхідно створити модель архітектури з використанням брокерів повідомлень та сховищ

даних з використанням хмарного середовища та порівняти існуючі підходи. Також слід порівняти принципи та методи .

Перейдемо до опису моделі дослідження в наступному розділі.

2 ОПИС ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

2.1. Порівняння існуючих методів для роботи з великими даними

На огляд візьмемо найбільш поширені характеристики ефективності для обчислення великих даних є:

- обробка даних в режимі реального часу;
- горизонтальна масштабованість;
- відмовостійкість;
- локальність даних;
- хмарне середовище розгортання.

Для дослідження цих метрик ефективності та способів застосування, необхідно описати існуючі варіанти побудови високонавантажених та широкомасштабних систем.

Зазвичай архітектуру великих даних використовують для аналізу, збору статистики, тощо. Для того щоби швидко обмінюватись даними в режимі реального часу, а потім їх проаналізувати, використовують брокери повідомлень. На сьогоднішній день існує велика кількість брокерів повідомлень, до популярних рішень можно віднести ActiveMQ, RabbitMQ та Kafka.

Брокер повідомлень працює за наступним принципом: виробник повідомлень передає повідомлення у чергу в серіалізованому форматі, на кожну чергу підписані користувачі, які вичитують повідомлення з черги та потім десеріалізують (див. Рис. 2.1). Цей підхід дозволяє скоротити час передачі даних за рахунок передачі повідомлень партією, коли досягається певна кількість повідомлень то тільки тоді повідомлення будуть надіслані в чергу де потім вони будуть зчитані.

Перше покоління брокерів повідомлень, таких як ActiveMQ, спиралося на парадигму MON (Message Orient Middleware).

Пізніше з'явилися гіперпродуктивні платформи обміну повідомленнями (їх часто називають потоковими процесорами), які більше підходять для парадигми потокового передавання. Два популярні засоби обробки потоків - це Apache Kafka та Amazon Kinesis Data Streams.

Порівнюючи усі існуючі брокери для повідомлень, Apache Kafka має велику перевагу над іншими конкурентами, цей брокер добре себе зарекомендував в архітектурі великих даних. Він працює досить швидко з великими об’ємами даних, що дозволяє здійснювати миттєву передачу повідомлень в режимі реального часу. Брокер розрахований на горизонтально масштабованість, тобто в залежності від завантаженості системи можна сконфігувати необхідну кількість вузлів для забезпечення обробки всіх наявних даних.

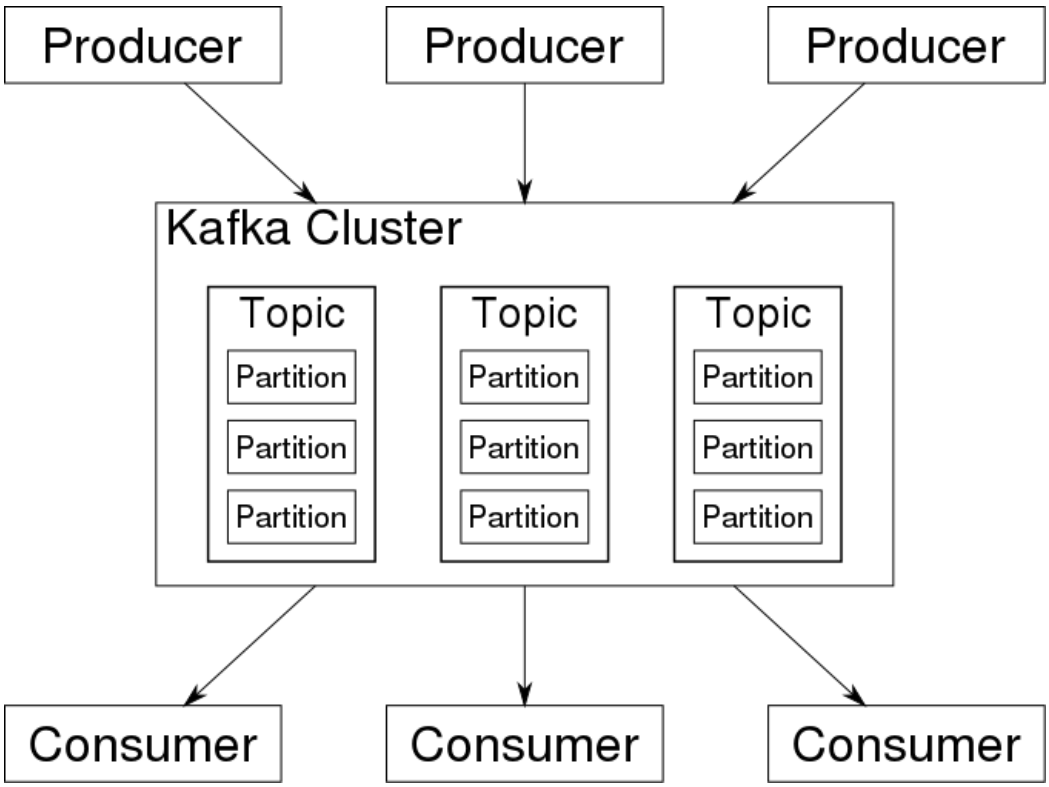

Рисунок 2.1 – Модель організації роботи брокера обміну повідомленнями

Архітектура великих даних потребує велику кількість обчислювальних ресурсів, дуже важливо мати можливість для горизонтальної та вертикальної

масштабованості бо кількість даних в теорії буде тільки зростати і потреба в додаткових обчислювальних пристроях буде вкрай необхідна. На етапі проектуванні дуже важливо взяти до уваги такі властивості для масштабування.

У сучасних хмарних сервісах існує багато інструментів для автоматичного розгортання серверів, сховищ даних, різноманітних утиліт та подальшого їхнього масштабування. Серед головних постачальників хмарних середовищ можна виділити Amazon AWS та Azure. Серед інструментів AWS досить багато сервісів для автоматичної масштабованості обчислюваних машин. Вони здатні стежити за показниками серверів та при великих навантаженнях вертикально або горизонтально масштабувати їх.

Серед всіх інструментів можемо виділити:

- AWS CloudFormation;
- AWS CloudWatch;
- AWS Autoscaling.

У AWS CloudFormation передбачена універсальна мова для моделювання і виділення ресурсів додатків AWS і сторонніх постачальників в хмарному середовищі. AWS CloudFormation дає можливість використати мови програмування або простий текстовий файл для автоматичного безпечного моделювання і виділення усіх ресурсів, необхідних для додатків по усіх регіонах і аккаунтах користувача. Таким способом отримується єдине джерело достовірної інформації про ресурси AWS і сторонніх постачальників.

Завдяки AWS CloudFormation можна моделювати усю інфраструктуру або текстового файлу, або мов програмування. Реєстр і інтерфейс командного рядка AWS CloudFormation спрощують управління ресурсами сторонніх постачальників за допомогою CloudFormation. Такий підхід дозволяє отримати єдине джерело достовірної інформації про усі ресурси і стандартизувати компоненти інфраструктури, які використовуються в організації, забезпечуючи відповідність конфігурації вимогам і прискорене усунення неполадок.

AWS CloudFormation виділяє ресурси додатків безпечним і відтворним чином, дозволяючи створювати і відтворювати інфраструктуру і додатки без необхідності виконувати ручні дії або писати власні скрипти. CloudFormation самостійно визначає, які операції слід виконувати при управлінні стеком, упорядковуючи їх найефективнішим чином, і автоматично відміняє зміни, якщо виявляються помилки.

Важливим компонентом будь-якої системи є процес запису подій в журнал, щоб мати можливість переглядати стан сервісів. В архітектурі великих даних необхідно забезпечити можливість збору записів до журналу подій з високої частотою. Серед популярних рішень, які підходять до даного типу архітектури, можемо виділити AWS CloudWatch Logs та ELK Stack.

Amazon CloudWatch - це сервіс моніторингу хмарних ресурсів AWS і додатків, працюючих на AWS. Можна використати Amazon CloudWatch для збору і відстежування метрик, накопичення і аналізу записів в журналі, а також для створення попереджень. Amazon CloudWatch може вести моніторинг ресурсів, призначених для користувача метрик додатків і сервісів, а також моніторинг будь-яких журналів додатків.

Amazon CloudWatch можна використати для отримання звітної інформації про систему, включаючи інформацію про використовувані ресурси, продуктивність додатків і загальної працездатності системи. Ці дані застосовуються для оперативного реагування і забезпечення стабільної роботи додатків. За допомогою CloudWatch Logs можна перевіряти свої журнали на наявність певних фраз, значень або шаблонів в режимі, близькому до реального часу. Наприклад, можна настроїти видачу попереджень про кількість помилок, зареєстрованих в системних журналах, або відображення графіків затримки веб-запитів, зареєстрованих в журналах додатків. Потім можна проглянути дані початкового журналу, щоб визначити джерело проблеми. Дані журналів можна зберігати необмежений час, використовуючи для цього недороге сховище з високою мірою надійності, щоб не позичати місце на жорсткому диску. При

цьому вони залишаються повністю доступними. CloudWatch Logs дозволяє відстежувати і зберігати журнали для аналізу роботи систем і додатків, а також для управління ними. Моніторинг додатків і систем в режимі реального часу. CloudWatch Logs можна використати для моніторингу додатків і систем за допомогою журналів. Оскільки при роботі з CloudWatch Logs для моніторингу використовуються дані існуючих журналів, ніяких змін в код вносити не вимагається.

Порівнюючи існуючі сервіси для збору та відображення записів в журналі, CloudWatch Logs має велику кількість переваг, до них можна віднести наступні пункти:

- використовуючи AWS сервіси для розгортання системи, є можливість налаштувати моніторинг за ресурсами;
- для запису записів в журнал не треба піднімти додаткові сервіси, все необхідне вже є зі старту;
- є можливість агрегувати даними на основі записів в журналі та відображати візуалізовані графіки;
- є можливість фільтрувати записи у журналах;
- існує можливість обирати журнал для створення запису.

Майже в кожній системі є потреба зберігати дані, в архітектурі великих даних це не є винятком а навіть навпаки. Необхідність в сховищах для зберігання та доступу до даних є дуже важливим елементом кожної системи, тому необхідно обрати доцільні інструменти для зберігання інформації, які б гарантували надійність та безпеку даних. Дуже важливим показником для обраного виду архітектури є швидкість читання даних, тому при виборі певних інструментів треба звертати на це певну увагу. Зважаючи на можливість стрімкого збільшення об’ємів даних, необхідно забезпечити можливість для масштабування сховищ даних та зберегти швидкість доступу до даних.

Найбільш доцільним до потреб зберігання даних в архітектурі великих даних є сервіс AWS S3.

AWS S3 - це сервіс зберігання об'єктів, що пропонує новітні підходи для вирішення будь-яких рішень, масштабованості, доступності і безпеки даних. Сервіс розрахований на зберігання і захист будь-яких об'ємів даних в різних ситуаціях, наприклад для забезпечення роботи сайтів, мобільних застосувань, для резервного копіювання і відновлення, архівації, облаштувань IoT і аналізу великих даних. AWS S3 надає прості у використанні інструменти адміністрування, які дозволяють організувати дані і точно настроїти обмеження доступу у відповідності потребами певної системи. AWS S3 забезпечує високу надійність і розрахованих на зберігання даних у великих об’ємах.

Можна легко збільшувати і скорочувати ресурси сховища відповідно до коливань потреб. Сервіс Amazon S3 гарантує надійність даних, оскільки він автоматично створює і зберігає копії усіх об'єктів з S3 у безлічі незалежних систем. Це означає, що дані доступні, коли вони потрібні, і захищені від збоїв, помилок і загроз.

AWS S3 забезпечує надійні можливості управління даними і відстежування пов'язаних з ними витрат, а також реплікації і захисту даних. Компонент S3 Access Points спрощує управління доступом до даних з певними дозволами для додатків, що використовують загальні набори даних. Можливості реплікації S3 дозволяють управляти реплікацією даних в межах регіону або в інших регіонах. Завдяки пакетним операціям S3 можна управляти великомасштабними змінами мільярдів об'єктів. Оскільки S3 працює з AWS Lambda, ви можете реєструвати дії, визначати попередження і автоматизувати робочі процеси без налаштування додаткової інфраструктури.

Сервіс дозволяє створювати масштабовані, надійні і безпечні рішення для резервного копіювання і відновлення за допомогою S3 і інших сервісів AWS, таких як S3 Glacier, Amazon EFS і Amazon EBS, щоб розширити або повністю замінити наявні локальні можливості. За допомогою AWS можна виконувати резервне копіювання даних, які вже знаходяться в хмарі AWS, або використати

AWS Storage Gateway, сервіс гібридного зберігання, для відправки резервних копій локальних даних в AWS.

В системі розрахованій на роботу з великими об’ємами інформації повстає питання як швидко та ефективно обробляти та аналізувати потоки інформації.

Архітектура великих даних стає актуальною та доцільною, коли йде робота з інформацією де один чи кілька обчислювальних машин не в змозі обробляти актуальні об’єми вхідної інформації за очікуваний відрізок часу. Для забезпечення продуктивної роботи з дійсно великими масивами інформації стають актуальними підходи великих даних.

Одним із основних підходів до роботи з великими даними є кластеризація. Маючи великі масиви даних необхідно розбити їх на пропорційні шматки та передати їх на обробку обчислювальним машинам, які будуть працювати паралельно та незалежно один від одного, де швидкість обчислення вхідних даних пропорційно залежить від кількості вузлів в кластері. Добре зарекомендував себе Apache Hadoop для вирішення задач з великим набором даних, яких реалізує головні принципи, методи, підходи роботи з великими даними.

Hadoop - це програмний проект, призначений для ефективної обробки великих наборів даних. Замість одного великого комп'ютера для обробки і зберігання даних Hadoop пропонує використати для паралельного аналізу величезних наборів даних кластери на базі стандартного апаратного забезпечення.

У сімействі Hadoop представлена безліч додатків і механізмів виконання, що пропонують різні інструменти для обробки аналітичних робочих навантажень. Сервіс AWS EMR дозволяє без зусиль створювати повністю налагоджені еластичні кластери екземплярів AWS EC2 для запуску Hadoop і інших застосувань, а також управляти цими кластерами.

Під Hadoop зазвичай розуміється проект Apache Hadoop, включаючи MapReduce (платформу виконання), YARN (менеджер ресурсів) і HDFS (розподілене сховище). Можна також встановити Apache Tez, платформу наступного покоління, для використання замість Hadoop MapReduce в якості

механізму виконання. Крім того, Amazon EMR включає файлову систему EMRFS, яка дозволяє Hadoop використати Amazon S3 в якості рівня зберігання.

Проте в сімействі Hadoop існують і інші застосування і платформи, включаючи інструменти для створення запитів з низькими затримками, графічні інтерфейси для створення інтерактивних запитів, різні інтерфейси на кшталт SQL і розподілені бази даних NoSQL. Сімейство Hadoop містить безліч інструментів з відкритим початковим кодом, призначених для розробки додаткових функціональних можливостей на базі основних компонентів Hadoop.

Механізми виконання Hadoop MapReduce і Tez, що входять в систему Hadoop, обробляють робочі навантаження за допомогою платформ, які розділяють завдання на дрібніші для розподілу між вузлами кластера Amazon EMR. Вони розроблені для забезпечення відмовостійкості з урахуванням того, що будь-який комп'ютер в кластері може у будь-який момент відмовити. Якщо сервер, що виконує завдання, вийде з ладу, Hadoop здійснить передачу цього завдання іншому комп'ютеру, і так до її виконання.

Можна швидко динамічно ініціалізувати новий кластер Hadoop або додати сервери в існуючий кластер Amazon EMR, значно понизивши час, що витрачається на забезпечення доступності ресурсів користувачам і фахівцям з обробки даних. Hadoop на платформі AWS дозволяє значно підвищити організаційну гнучкість за рахунок скорочення витрат і часу, яке витрачається на розподіл ресурсів для експериментів і розробки.

Hadoop в Amazon EMR дозволяє гнучко запускати кластери у будь-якій кількості зон доступності у будь-якому регіоні AWS. Потенційну проблему або загрозу, існуючу в одному з регіонів, можна без зусиль обійти за декілька хвилин за допомогою запуску кластера в іншій зоні.

Планування ресурсів до розгортання середовища Hadoop часто обертається простоєм дорогого устаткування або нестачею потужностей. За допомогою Amazon EMR можна створювати кластери з потрібними ресурсами за лічені

хвилини і використати Auto Scaling для динамічного масштабування вузлів у бік збільшення і зменшення.

Завдяки своїм властивостям масштабованості Hadoop

зазвичай використовується для обробки великих даних. Для підвищення продуктивності обробки в кластер Hadoop можна додати додаткові сервери з необхідними об'ємами ресурсів ЦП і пам'яті.

Hadoop пропонує високий рівень надійності і доступності при паралельній обробці обчислювальних аналітичних робочих навантажень. Завдяки поєднанню високої доступності, надійності і масштабованості процесу обробки Hadoop є найкращим рішенням для роботи з великими даними. Сервіс Amazon EMR дозволяє за декілька хвилин створити і настроїти кластер екземплярів Amazon EC2 для запуску Hadoop і почати отримувати користь зі своїх даних.

3 РЕЗУЛЬТАТИ ТЕОРЕТИЧНИХ ДОСЛІДЖЕНЬ

3.1. Проектування моделі архітектури великих даних

Дослідивши головні підходи та принципи великих даних, можна перейти до моделювання архітектури великих даних. Існує багато працюючих систем в різних сферах життєдіяльності, деякі з них потребують обчислень досить великих за об’ємами масивів даних, наприклад для аналізу статистики, поведінки користувачів, тощо, де доцільно буде використовувати підходи для обробки великих даних. Навіть в існуючі системи, які були спроектовані без урахування потреб роботи з великими даними, з досить не великими змінами до них буде не важко інтегрувати в них частину архітектури великих даних. Необхідно буде лише забезпечити передачу певних даних для подальшого їхнього аналізу.

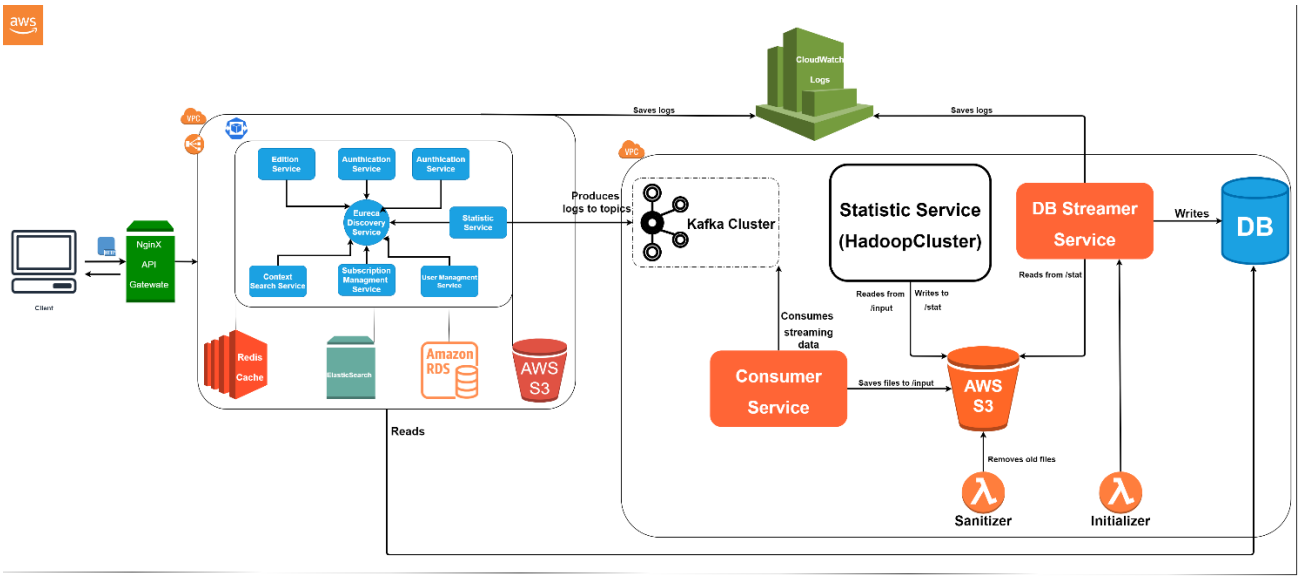

Рисунок 3.1 – Модель архітектури великих даних

На рисунку 3.1 зображено реалізація архітектури великих даних, тип даної архітектури називається як Каппа-архітектура, точкою входу даних виступає Apache Kafka. Мікро сервіс “Statistics Service” надсилає повідомлення,

які несуть в собі інформацію для подальшого аналізу та відображення статистики, у серіалізованому вигляді до брокера повідомлень Apache Kafka, які в свою чергу підхоплюються сервісом “Consumer Service”.

“Consumer Service” підписаний на канали Kafka, сервіс споживає повідомлення та записує їх в файл, файли зберігаються в ресурсах AWS EC2, раз на годину ініціалізується скрипт, який завантажує всі накопичені за цей проміжок часу файли на AWS S3 в папку “/input”. Для оптимізації процесу передачі повідомленнями є можливість сконфігувати клієнта, який відправляє або приймає повідомлення від Apache Kafka, в конфігурації вказується кількість повідомлень в партії для однієї передачі, час який буде зберігатися повідомлення та інші, все це налаштовується емпіричним шляхом, де все залежить від інтенсивності надходження інформації. Варто відзначити що обраний брокер обміну повідомленнями здатен розподіляти навантаження між своїми вузлами в кластері, що дозволяє швидко та без жодних затримок обмінюватись повідомленнями.

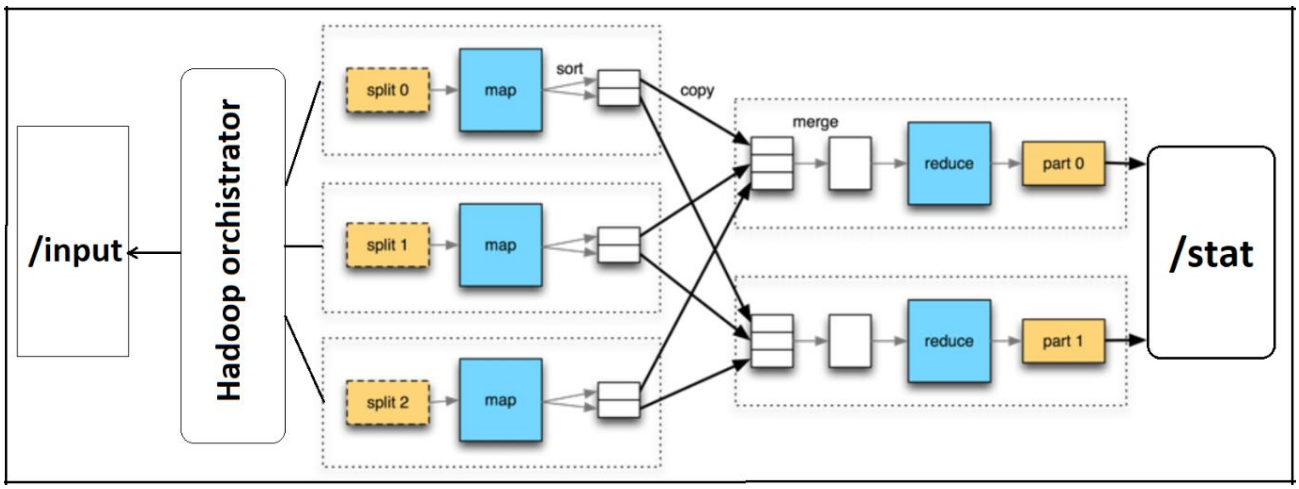

Рисунок 3.2 – Модель роботи кластера Hadoop

По мірі накопичення певної кількості необроблених файлів в папці “/input”, “Statistic Service” починає завантажувати всі файли за певний проміжок часу. Hadoop сервіс рівномірно розподіляє завантажені файли для подальшого

обчислення в окремі потоки, де дані будуть в подальшому обчислені. Statistic Service повністю реалізує модель MapReduce (див. рис. 3.2), де дані проходять всі необхідні етапи обчислень, після обробки всіх даних результат обчислень записується в нові файли та партіями завантажуються в AWS S3 в папку “/stat”. Hadoop MapReduce досягає дуже ефективних показників при обчислені даних, це досягається за допомогою розпаралелення обчислень на окремі незалежні задачі, де для кожної задачі буде виділена окрема машина для обчислень, фреймворк в автоматичному режимі повністю контролює життєвий цикл створених потоків та утилізацію потоків.

Останньою ланкою в побудованій Каппа-архітектурі є “DB Streamer Service”, який завантажує партіями файли з вихідною інформацією з папки “/stat”. Цей сервіс зберігає дані з файлів, які були записані “Statistic Service”, в базу даних. “DB Streamer Service” вичитує інформацію з файлів та зберігає її у відповідні таблиці, зберігання до БД відбувається партіями, беручи до уваги те що даних величезна кількість, роботи зберігання по одному дуже затратно, тому найоптимальнішим рішенням є зберігати інформацію великими партіями, цей підхід менше навантажує СУБД та робить процес збереження набагато швидшим. Сервіс працює тільки коли в цьому є необхідність, процес обчислення інформації може бути достатньо довгим, а для ефективної роботи необхідно мати достатню кількість файлів з обчисленою інформацією. “DB Streamer Service” запускається лямбда сервісом “Initializer”, який періодично відслідковує AWS S3, коли досягається необхідна кількість файлів потребуючих зберігання відбувається запуск зберігання даних. У разі не достатньої кількості файлів для запуску “DB Streamer Service” сервіс все одно буде запущеним, це відбувається раз на годину. Цей механізм допомагає зменшити використання обчислювальних ресурсів, тобто сервер працює тільки за необхідність.

Назбиравши за деякий час інформацію, в сховищі накопичується досить велика кількість файлів, які вже буди обробленими та зміст яких був вже збережений до БД, для видалення застарілих файлів використовується лямбда

сервіс “Sanitizer”, який час від часу виконує очищення більше не потрібних файлів, сервіс стежить за вмістом сховища та видаляє файли які були опрацьовані в попередні дні.

Невід’ємною частиною кожної системи є вимога переглядати записи в журналі подій. В побудованій архітектурі було використано сервіс CloudWhatch від AWS, крім перегляду подій є можливість агрегувати вмістом записів в журналі та виводити потрібну інформацію, сервіс в автоматичному режимі будує гістограми або графіки (див. рис.3.3).

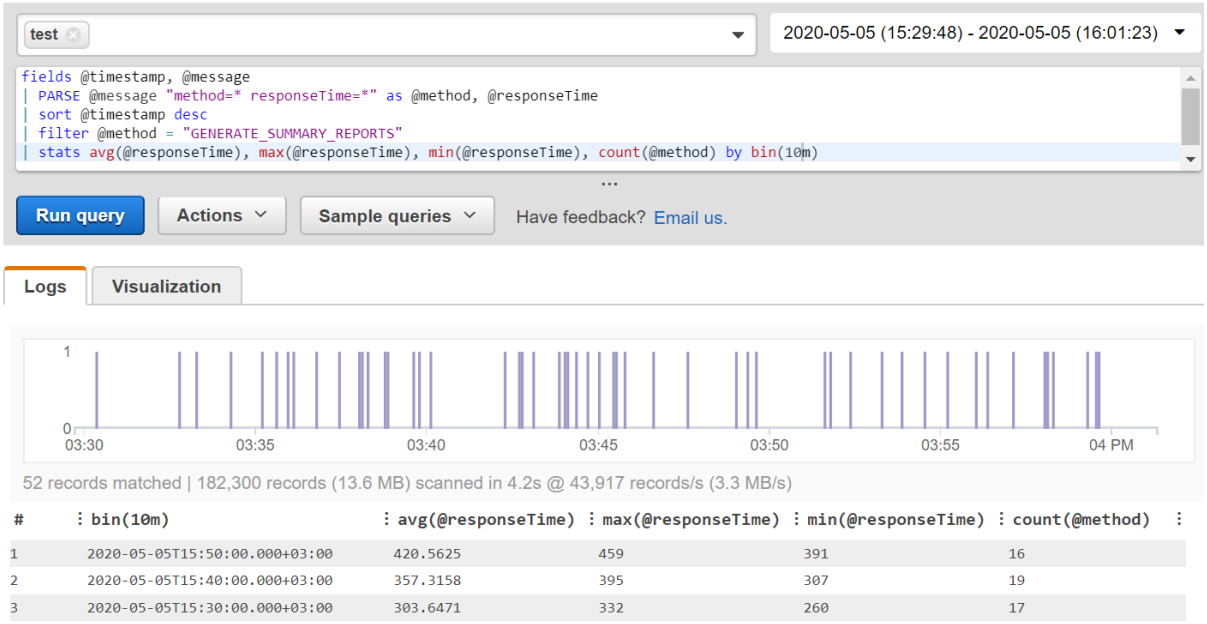

Рисунок 3.3 – Відображення записів в журналі CloudWatch

CloudWatch має специфічну мову запитів, яка дозволяє розбирати запис на окремі поля, які в подальшому можна використати для побудови гістограми або графіку (див. рис. 3.4). Серед інших можливостей цієї мови запитів також є фільтрація, сортування, групування за часом та ліміт на кількість відображення подій. За допомогою цієї мови запитів вдалося розібрати записи подібного формату ”method=GENERATE_SUMMARY_REPORTS latency=263” та за рахунок отриманих даних обчислити показники виконання певних функціональних елементів системи.

Крім того CloudWatch здатен слідкувати за станом обчислювальних машин та в разі критичних ситуацій може надсилати повідомлення на електронну пошту. За необхідністю записи з журналу можна експортувати в сховище AWS S3, для архівування.

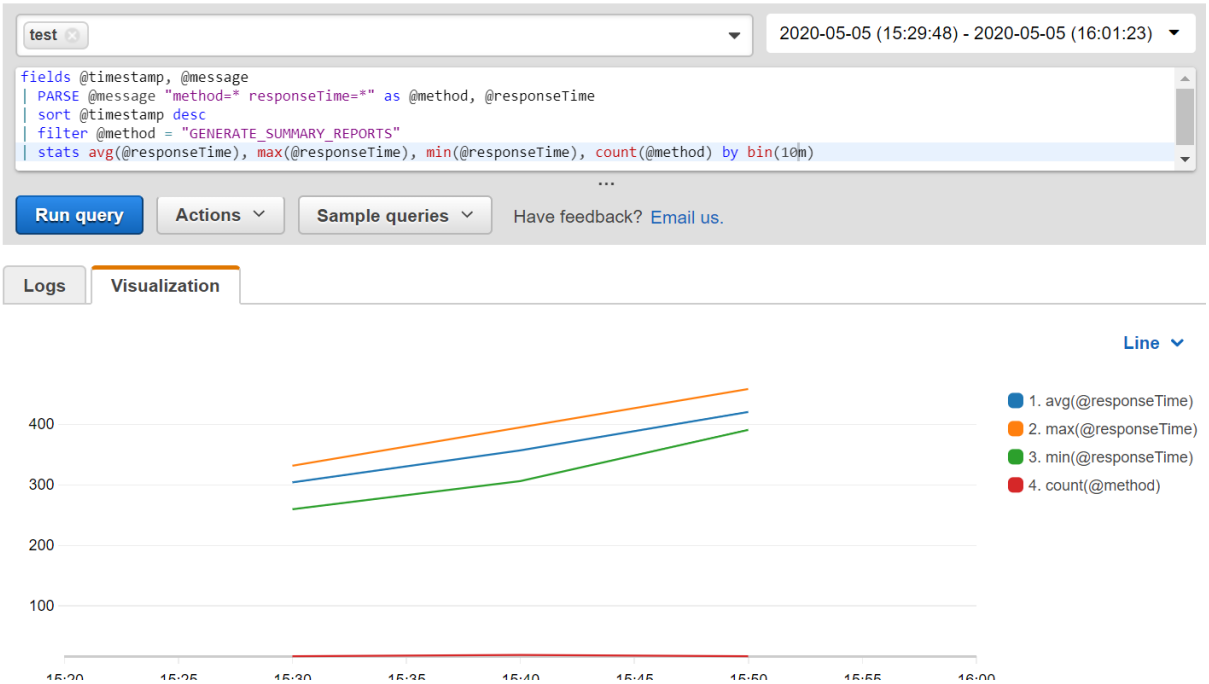

Рисунок 3.4 – Візуалізація метрик в журналі подій CloudWatch

В побудованій архітектурі вимога до стабільної роботи серверів вирішена за допомогою розподілення обчислювальних задач по різним EC2 екземплярам. Кожний вид сервісів має по декілька екземплярів, де кількість залежить від завантаженості системи, функцію розподілення ресурсів вирішує AWS Load Balancer.

Для забезпечення безпеки даних всі критичні елементи системи працюють в віртуальній приватній хмарі, це зменшує до нуля шанси зловмисників заволодіти приватною інформацією, яка зберігається в базі даних та сховищі. Отримати доступ до обчислювальних машин можна лише за допомогою секретних ключів доступу. Для доступу до захищених серверів, клієнтські додатки ведуть своє спілкування з API Gateway, в побудованій архітектурі використовується NGINX,

який виконую вище зазначену роль. API Gateway містить файл з конфігураціями, де прописуються необхідні сертифікати, порти та посилання, тобто вказуються перелік ресурсів до якого буде мати клієнт. До приватної хмари доступ мають лише ті кому дозволено його мати, це вирішує проблему несанкціонованого доступу до інформації. Весь існуючий API в системі стає захищеним, мобільному або веб клієнту відкривається список доступних йому ресурсів, а всі інші залишаються не досяжними за межами приватної хмари. Цей підхід має ряд переваг, зовсім не обов'язковим в такому підході стає аутентифікація для деяких ресурсів, бо доступ до них мають лише ресурси з тієї ж хмари.

Маючи велику інфраструктуру, стає все важче її піднімати, тож доцільним в такому випадку автоматизувати процес підняття потрібних для функціювання системи ресурсів.

```
1  Ec2Instance:
2    Type: AWS::EC2::Instance
3    Properties:
4      ImageId:
5        Fn::FindInMap:
6          - "RegionMap"
7          - Ref: "AWS::Region"
8          - "AMI"
9      KeyName:
10       Ref: "KeyName"
11     NetworkInterfaces:
12       - AssociatePublicIpAddress: "true"
13         DeviceIndex: "0"
14         GroupSet:
15           - Ref: "myVPCEC2SecurityGroup"
16         SubnetId:
17           Ref: "PublicSubnet"
```

Рисунок 3.5 – Кофігурація для розгортання серверу

За допомогою AWS CloudFormation можливо в декларативному форматі описати конфігурацію для створення певних ресурсів (див. рис. 3.5). Сервіс пропонує можливість створення інфраструктури описаного в форматі YAML або JSON, за допомогою CloudFormation було сгенеровано EC2 сервера для деяких сервісів та Lambda сервіси.

ВИСНОВКИ

В рамках роботи було проведено дослідження методів та принципів роботи з великими даними та досліджено підходи розгортання необхідної інфраструктури в хмарному середовищі. Було здійснено аналіз можливих видів архітектури великих даних. Проаналізовано інструменти та підходи розгортання системи в хмарному середовищі та підготовки необхідної інфраструктури для забезпечення повноцінного її функціювання.

На підставі результатів дослідження було спроектовано архітектуру великих даних відповідно до поставлених вимог. Було розроблено модель архітектури з можливостями горизонтальної та вертикальної масштабованості системи, та створено механізм для забезпечення стабільної роботи незалежно від завантаженості системи.

Було ураховано проблему безпеки зберігання та захисту інформації від зовнішніх загроз за рахунок використання приватних хмар, доступ до яких здійснюється тільки за допомогою ключів доступу. Було створено механізми для доцільного використання ресурсів хмарного обчислення, як очищення сховищ від не потрібної інформації та керування запусками серверів тільки в разі потреби для зменшення споживання ресурсів. Інтегровано інструменти для розробників для моніторингу стану системи та використання нею ж хмарних ресурсів, що дозволяє в разі несподіваних ситуацій вчасно знати про існування певних проблем та швидкого їх вирішення.

Поєднання новітніх технологій великих даних та хмарного обчислення дозволяє створювати системи здатні обробляти запити мільйонів користувачів по всій земній кулі, що дає можливість створювати будь-які системи, котрі в недалекому минулому ще досі здавалися не досяжними для людства.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Shiva Jegan R.D, Vasudevan S.K, Abarna K., Prakash P, Srivathsan S., Gangothri V., "Cloud computing: A Technical Gawk". 2014. URL: https://www.amrita.edu/publication/cloud-computing-technical-gawk.

2. Pedro Caldeira Neves, Jorge Bernardino, "Big Data in the cloud: A survey". 2015. URL: https://www.ronpub.com/OJBD_2015v1i2n02_Neves.pdf .

3. Sourav Kunal, Arijit Saha, Ruhul Amin, “An overview of cloud-fog computing: Architectures, applications with security challenges”. 2019. URL: https://www.researchgate.net/publication/333162093.

4. Cao, N., Wang, C., Li, M., Ren, K., Lou, W., “Privacy-preserving multi-keyword ranked search over encrypted cloud data.”. 2014. URL: https://pdfs.semanticscholar.org/cbe3/e7cf2c1f64ce667cacad6666e8f62fa2ff3f.pdf.

5. Viraj G. Mandlekar, Viresh Kumar Mahale, “Survey on Fog Computing Mitigating Data Theft Attacks in Cloud”. 2014. URL: https://www.ijircst.org/DOC/3_IRP2144caa8e0e-a940-4c1d-81c1-5286717307d3.pdf.

6. Cash, D., et al. “Dynamic searchable encryption in very-large databases: Data structures and implementation”. 2014. URL: https://eprint.iacr.org/2014/853.pdf.

7. Dsouza, C., Ahn, G.J., Taguinod, M. “Policy-driven security management for fog computing: Preliminary framework and a case study”. 2014. URL: https://sefcom.asu.edu/publications/Policy-driven-security-management-iri2014.pdf.

8. Pedro GL et al. “Edge-centric computing: vision and challenges”. 2015. URL: http://disi.unitn.it/~montreso/pubs/papers/ccr15.pdf.

9. Rajkumar B., Rodrigo C., “Big Data Principles and Paradigms”. 2016. URL: http://dhoto.lecturer.pens.ac.id/lecture_notes/internet_of_things/Big%20Data%20Principles%20and%20Paradigms.pdf.

10. Evangelos Pournaras, “Big Data Analytics.”. 2017. URL: https://ethz.ch/content/dam/ethz/special-interest/gess/computational-social-science-dam/documents/education/Spring2017/Data_science/course4.pdf

11. Гвоздинский А.Н., Губин В.А., Юрдига Л. А. Кластеризация слабоструктурированных текстовых документов // ХНУРЭ: электронная версия научно-технического журнала 2011. № 1 URL: http://openarchive.nure.ua/bitstream/document/1784/1/RI_2011_1-044-047.pdf (дата звернення: 17.05.2020).